Metabarcoding

Dr. Sammy Wambua

12th July, 2022

DADA2 background

This tutorial will demonstrate microbial characterization utilizing 16S rRNA gene sequences generated with Illumina’s MiSeq platform using paired end reads. It is a walkthrough of the DADA2 pipeline adopted from the author’s original tutorial which may be accessed **here** (https://benjjneb.github.io/dada2/tutorial.html).

The dataset to be used is small enough to run on a laptop. It is generated by a laboratory investigating the effect of gut microbiome on mice health. Microbial sequences from fecal samples collected daily from one animal for 365 days post weaning (dpw). Since a rapid change of weight is observed during the first 10 dpw, we can do microbiome comparison between this time and, say, after 100 dpw to see if the microbial communities differ between early and late dpw. Getting Started DADA2 runs on R producing an amplicon sequence variant (ASV) table (equivalent of the traditional OUT table) which records the number of ASVs observed in each sample and is used for taxonomy assignment. Further analysis and visualization requires importation to **phyloseq** (https://joey711.github.io/phyloseq/) and use of other R packages. In all, you need install the following packages in R:

```
library(dada2); packageVersion("dada2")
BiocManager::install("phyloseq")
BiocManager::install("biostrings")
BiocManager::install("decipher")
```

Incase `BiocManager` has not been installed yet on your R (something that should only be done once), install it before installing the packages.

```
chooseCRANmirror()
install.packages("BiocManager")
```

Also install the following:

```
install.packages("tidyverse")
install.packages("vegan")
```

This step should be performed outside of R

Download sequences zip folder http://www.mothur.org/w/images/d/d6/MiSeqSOPData.zip (http://www.mothur.org/w/images/d/d6/MiSeqSOPData.zip), unzip it and place the folder “MiSeq_SOP” in your working directory.

Download these files too and place them in your working directory:
https://zenodo.org/record/3731176/files/silva_nr_v138_train_set.fa.gz
(https://zenodo.org/record/3731176/files/silva_nr_v138_train_set.fa.gz)
https://zenodo.org/record/3731176/files/silva_species_assignment_v138.fa.gz
(https://zenodo.org/record/3731176/files/silva_species_assignment_v138.fa.gz)

You may use the following scripts if you are working on the terminal:

```
wget http://www.mothur.org/w/images/d/d6/MiSeqSOPData.zip

unzip MiSeqSOPData.zip

rm -r __MACOSX/

cd MiSeq_SOP

wget https://zenodo.org/record/3731176/files/silva_nr_v138_train_set.fa.gz

wget https://zenodo.org/record/3731176/files/silva_species_assignment_v138.fa.gz

cd ..
```

In R, confirm that you downloaded the sequences. You should see a total of 48 files:

```
path <- "~/MiSeq_SOP" # change to the directory containing the fastq files

list.files(path)
```

Quality Filtering and Trimming

Sort to get matched lists of forward and reverse files

```
fnFs <- sort(list.files(path, pattern="_R1_001.fastq", full.names = TRUE))

fnRs <- sort(list.files(path, pattern="_R2_001.fastq", full.names = TRUE))
```

Extract sample names from the first element for the file names

```
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

Inspect read quality profiles

```
plotQualityProfile(fnFs[1:2])

plotQualityProfile(fnRs[1:2])
```

Feel free to play with the sub-setting to see more sample profiles. What do you think of the quality profiles?

Define a subdirectory where filtered reads will be placed

```
filtFs <- file.path(path, "filtered", paste0(sample.names, "_F_filt.fastq.gz"))

filtRs <- file.path(path, "filtered", paste0(sample.names, "_R_filt.fastq.gz"))

names(filtFs) <- sample.names

names(filtRs) <- sample.names
```

We’ll use standard filtering parameters

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen=c(240,160), maxN=0, maxEE=c(2,
2), truncQ=2, rm.phix=TRUE, compress=TRUE, multithread=TRUE) # On Windows, multithread=F
ALSE head(out)
```

Learn the Error Rates

The DADA2 algorithm uses a parametric error model (`err`) and every amplicon dataset has a different set of error rates. The `learnErrors` method learns this error model from the data. This step takes 3 minutes to run on a 2013 Macbook Pro but can take much longer in other machines.

```
errF <- learnErrors(filtFs, multithread=TRUE)
```

```
errR <- learnErrors(filtRs, multithread=TRUE)
```

```
plotErrors(errF, nominalQ=TRUE)
```

Study the plots. The error rates for each possible transition (A→C, A→G, …) are shown. Points are the observed error rates for each consensus quality score. The black line shows the estimated error rates after convergence of the machine-learning algorithm. The red line shows the error rates expected under the nominal definition of the Q-score. Here the estimated error rates (black line) are a good fit to the observed rates (points), and the error rates drop with increased quality as expected.

Sample Inference

Dereplicate to get unique sequences and save on computation time downstream.

```
derep_forward <- derepFastq(filtFs, verbose=TRUE)
```

```
derep_reverse <- derepFastq(filtRs, verbose=TRUE)
```

name the derep-class objects by the sample names

```
names(derep_forward) <- sample.names

names(derep_reverse) <- sample.names
```

Apply the core sequence-variant inference algorithm to the dereplicated sequences.

```
dadaFs <- dada(derep_forward, err=errF, multithread=TRUE)

dadaRs <- dada(derep_reverse, err=errR, multithread=TRUE)
```

How many sequence variants were inferred?

```
dadaFs[[1]]
```

In the first sample, 128 true sequence variants were inferred from the 1979 unique sequences. Merge the forward and reverse reads together to obtain the full denoised sequences. Merging is performed by aligning the denoised forward reads with the reverse-complement of the corresponding denoised reverse reads, and then constructing the merged “contig” sequences.

```
merged_reads <- mergePairs(dadaFs, derep_forward, dadaRs, derep_reverse, verbose=TRUE)
```

Construct sequence table

We can now construct an amplicon sequence variant table (ASV) table, a higher-resolution version of the OTU table produced by traditional methods.

```
seqtab <- makeSequenceTable(merged_reads)
```

You may inspect the ASV table:

```
dim(seqtab)
```

```
table(nchar(getSequences(seqtab)))
```

Remove Chimeras

The DADA2 method used corrects substitutions and indel errors, but chimeras remain. These are easily identifiable after denoising and are removed with:

```
seqtab.nochim <- removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE, verbos
e=TRUE)

dim(seqtab.nochim)
```

What percentage of our reads did we keep after removing chimeras?

```
sum(seqtab.nochim)/sum(seqtab)
```

How about from the start?

```
getN <- function(x) sum(getUniques(x))

track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN), sapply(merged_reads, get
N), rowSums(seqtab.nochim))

colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")

rownames(track) <- sample.names

head(track)
```

Assign taxonomy

The authors of DADA2 maintain formatted fastas for the RDP, GreenGenes clustered at 97% identity, and the Silva reference databases for taxonomy assignment. In this tutorial we will use the latest Silva reference database version 138 which you downloaded in the start. (Actually confirm if this is still the latest, and adapt your codes accordingly!!).

```
taxa <- assignTaxonomy(seqtab.nochim, "~/silva_nr_v138_train_set.fa.gz", multithread=TRU
E)
```

```
taxa <- addSpecies(taxa, "./silva_species_assignment_v138.fa.gz")
```

Inspect the taxonomic assignments

```
taxa.print <- taxa # Removing sequence rownames for display only

rownames(taxa.print) <- NULL

head(taxa.print)
```

Nonesensical assignments eg Eukaryota NA NA NA NA NA, are an indication that your reads may be in the opposite orientation as the reference database. Tell dada2 to try the reverse-complement orientation with `assignTaxonomy(…, tryRC=TRUE)` and see if this improves the assignments.

You may want to save the taxonomy assigned sequence table now, with `saveRDS(taxa, "~/name.rds")`, to avoid re-doing the process from the start upon discontinuing the R session! Upon resuming R session, you can restart processing your data from this point by reading the saved object into a variable i.e. `variable <- readRDS("name.rds")`.

Analysis and Visualization

Downstream analysis and visualization of the microbiome data so far processed requires use of other R packages once the data has been imported into a phyloseq object. The phyloseq R package is a powerful framework for analysis of microbiome data.

Construct a sample `data.frame`. We will do this using the information encoded in the filenames but ideally this would be done by reading a sample data in from a file (metadata).

```
samples.out <- rownames(seqtab.nochim)
subject <- sapply(strsplit(samples.out, 'D'), `[`, 1)
gender <- substr(subject, 1, 1)
subject <- substr(subject, 2, 999)
day <- as.integer(sapply(strsplit(samples.out, 'D'), `[`, 2))
samdf <- data.frame(Subject=subject, Gender=gender, Day=day)
samdf$When <- "Early"
samdf$When[samdf$Day>100] <- 'Late'
rownames(samdf) <- samples.out
```

Construct a phyloseq object

```
ps <- phyloseq(otu_table(seqtab.nochim, taxa_are_rows = FALSE), sample_data(samdf), tax_
table(taxa))

ps <- prune_samples(sample_names(ps) !='Mock', ps)
```

Store DNA sequences of ASVs in the refseq slot of the phyloseq object and rename taxa to a short string.

```
dna <- Biostrings::DNAStringSet(taxa_names(ps))
names(dna) <- taxa_names(ps)
names(dna) <- merge_phyloseq(ps, dna)
taxa_names(ps) <- paste0("ASV", seq(ntaxa(ps)))
```

See what constitutes the phyloseq object

```
ps
```

It is possible to also include a phylogenetic tree in this object. You may want to save this object with `saveRDS(ps, "~/ps.rds")` as it contains all your data concatenated.

Before inspecting the different diversity metrics, you may want to confirm that your samples were sequenced to a sufficient depth by quickly plotting a simple rarefaction curve with the vegan package.

```
rarecurve(otu_table(ps), step = 100, ylab = "Observed ASVs", xlab = "Number of reads", m
ain = "Rarefaction curves", label = FALSE))
```

The `rarecurve` function has a number of other options that you may play with to improve on the presentation of your curves. These can be assessed under the help menu (`?rarecurve`).

We can now plot a number of metrics and make the comparisons we set out to.

Alpha diversity

```
plot_richness(ps, x="Day", measures=c("Shannon", "Simpson"), color="When")
```

Beta diversity

```
ps.prop <- transform_sample_counts(ps, function(otu) otu/sum(otu))

ord.nmds.bray <- ordinate(ps.prop, method="NMDS", distance="bray")

plot_ordination(ps.prop, ord.nmds.bray, color="When", title="Bray NMDS")
```

Bar plot

We can plot the taxonomic distribution of the top 20 sequences

```
top20 <- names(sort(taxa_sums(ps), decreasing=TRUE))[1:20]
ps.top20 <- transform_sample_counts(ps, function(OTU) OTU/sum(OTU))
ps.top20 <- prune_taxa(top20, ps.top20)
plot_bar(ps.top20, x="Day", fill="Family") + facet_wrap(~When, scales="free_x")
```